Using Conceptual Graph Theory to Support Schema Integration

Paul Johannesson

Department of Computer and Systems Sciences
Stockholm University
Electrum 230, S-164 40 Kista, Sweden
email: pajo@sisu.se

Abstract. Two major problems in schema integration are to identify correspondences between different conceptual schemas and to verify that the proposed correspondences are consistent with the semantics of the schemas. This problem can only be effectively addressed if the conceptual schema is expressed in a semantically rich modelling formalism. We introduce such a modelling formalism, the distinguishing feature of which is the use of case grammar. We show that it is easier to identify correspondences between schemas expressed in this formalism than in schemas formulated in traditional modelling languages. The main reason for this is that case grammar standardizes the terminology in conceptual schemas by providing a set of meaningful and established labels for conceptual relations.

1 Introduction

Interoperability is becoming a critical issue for many organisations today. An increasing dependence and cooperation between organisations has created a need for many enterprises to access remote as well as local information sources. Further, even a single enterprise may have several independent information bases as a result of departmental autonomy. Thus, for many organisations it becomes important to be able to interconnect existing, possibly heterogeneous, information systems. An essential part of this activity is database integration, i.e. the process of constructing a global schema from a collection of existing databases. Database integration is similar to view integration, a process in classical database design, which derives an integrated schema from a set of user views. A generic term, *schema integration* [Batini86], has been introduced to refer to both processes.

The schema integration process can be divided into three phases: schema comparison, schema conforming, and schema merging. *Schema comparison* involves analysing and comparing schemas in order to determine correspondences, in particular different representations of the same concepts. *Schema conforming* involves modifying one or both of the schemas to be integrated until each phenomenon in the Universe of Discourse (UoD) is represented in the same way in both schemas. *Schema merging* involves superimposing the schemas in order to obtain one integrated schema.

The most difficult part of the schema integration process is schema comparison. The fundamental problem here is that a UoD can be modelled in many different ways. The same phenomenon can be seen from different levels of abstraction, or represented using different properties. More precisely, three types of differences between schemas modelling the same UoD can be identified: differences in terminology, differences in structure, and differences in focus. Differences in terminology arise, as people from different organisations, or from different areas of the same organisation, often refer to the same things using their own terminology. The problematic relationships between terms are of two types: synonyms and homonyms. Synonyms occur when the same object or relationship in the UoD is represented by different names in the two schemas. Homonyms occur when different objects or relationships are represented by the same name in the two schemas. Differences in structure arise when the same aspects of a UoD have been modelled using different constructs. As an example, in a conceptual schema we can classify objects into categories in two different ways. Either a number of subtypes of a given object type can be introduced, or an attribute can be used that has a value indicating the category to which an object belongs. Differences in focus occur when two schemas describe the same UoD but bring different aspects of it into focus. For example, two schemas may both model persons, but one schema may focus on the physical properties of persons, such as weight, length, and hair colour, whereas the other schema may describe the social characteristics of a person, such as language, education, and profession.

The types of schema differences identified above make it difficult to compare schemas. One way to alleviate this problem is to use a modelling formalism that is semantically richer than traditional semantic data models. The main contribution of this paper is to show how a modelling formalism based on conceptual graph theory [Sowa84] can be used to facilitate the comparison step in the schema integration process. We claim that conceptual graph theory provides a means for reducing schema dissimilarity caused by differences in terminology and focus. In the next section, we briefly review related research. In section 3, we describe a modelling formalism based on conceptual graph theory. In section 4, we show how schema correspondences can be expressed formally and give conditions for determining when two schemas can be meaningfully integrated. In section 5, we show how conceptual graph theory can be utilized to facilitate schema comparison, and in the final section, we summarize the paper and present some conclusions.

2 Related Research

Database management systems have been available for more than two decades, mainly in the form of the hierarchical, network, and relational models. In the mid 1970s the development of conceptual modelling approaches (also called semantic data models) was initiated. These were introduced primarily as schema design tools, meaning that a schema should first be designed in a high level conceptual model and then translated into one of the traditional models for implementation. The focus of the semantic models was to accurately model data relationships that arise frequently in database applications. One of the first and most influential semantic data models was Chen's Entity-Relationship (ER) approach, which was proposed in 1976, [Chen76]. The field of conceptual modelling has continued to evolve, trying to find models capable of expressing more of the semantics of an application, including its behavioural aspects. Several new abstraction mechanisms have been proposed, such as specialization, aggregation, and association. Some of the most well-known conceptual modelling approaches are the Semantic Data Model, SDM [Hammer81], the Entity Category Relationship model, [ElMasri85], and NIAM, [Nijssen89]; for comprehensive surveys, see [Hull87] and [Peckham88]. In its search for more expressive formalisms, the semantic modelling area has been heavily influenced by logic and logic programming, [Gallaire84], [Lloyd87]. In this paper, we formalize a conceptual modelling approach using first order logic.

Research in the area of schema integration has been carried out since the beginning of the 1980s. A comprehensive survey of the area can be found in [Batini86]. Most of the work has been done in the context of the relational model [Biskup86], the functional model [Motro87], or (some extended version of) the ER model, [Larson89], [Spaccapietra92], [Johannesson93a], [Johannesson93b], [Johannesson93c]. Most work has focused on how to merge a number of schemas given a set of proposed schema correspondences. In recent years, however, several researchers have also shown an interest in heuristic methods for identifying equivalences and similarities between schemas. In [Bouzhegoub90], constructs from different schemas are compared by means of a unification process based on similarities between names, structures, and constraints. The paper [Song92] proposes a method for identifying schema similarities, which utilizes a semantic dictionary and a taxonomy structure. Another idea is that of using linguistic tools, such as a thesaurus, and information retrieval theory to build a global data structure that helps automate the schema comparison process. Bright and Hurson report on some results obtained from this approach in [Bright90]. Collet, Huhns, and Shen [Collet91] have proposed a method for integrating heterogeneous information systems, which uses the Cyc [Guha90] knowledge base as a global schema to resolve inconsistencies.

3 Conceptual Schemas

An essential step in the development of an information system is to construct a conceptual model of a UoD. The conceptual model is to provide an abstract, implementation independent description of the UoD according to the conceptualisation principle [ISO82]. This principle states that the conceptual model must not take those aspects into account that are computer related, such as data representation, physical data organisation, or message formats.

Most approaches to conceptual modelling, e.g. the ER and NIAM models, restrict their attention to the static (or data) aspects of a UoD. The basic concept in all modelling approaches is the object (entity); objects judged as being similar are grouped together into object types, such as Employee and Department. Associations between objects are represented by means of attributes and/or relationships. It is usually possible to specify constraints that express propositions which should hold true in each state of the UoD. In most modelling approaches only certain simple types of constraints can be expressed, such as cardinality constraints. Recently, however, a few modelling languages have been developed which provide means for specifying more general types of constraints.

A limitation of traditional conceptual modelling approaches is that constraints are the only means of formally specifying the semantics of a schema. Consequently, much of the semantics has to be expressed by informal descriptions given in natural language and by an appropriate labelling of object types and attributes. One way to overcome this problem is to use a theory based on linguistics as the foundation of a formal language for expressing conceptual models. In a paper by Creasy and Moulin, [Creasy92], conceptual graph theory is proposed as the basis for a conceptual modelling language. Conceptual graphs provide a theory for representing knowledge structures at a conceptual level with richer semantics based on results from AI, linguistics, and logic, see [Sowa84]. Below, we will first give a brief and informal overview of conceptual graph theory. We will then show how this theory can be extended by language constructs for specifying constraints. Finally, we will present a formalization of the resulting modelling language.

Concepts are defined as the representations of objects in the UoD. A concept is characterized by two elements. First, a type representing some category, for example Human, Animal, or Employee. Secondly, a referent (object identifier) that denotes a specific object. Using a graphical notation, a concept type is represented by a rectangular box. Concept types are organized in a generalization hierarchy; we write C1 £ C2 if C1 is a specialization of C2. An example of a small generalization hierarchy is WOMAN £

HUMAN £ ANIMAL £ LIVING THING. An association, or elementary link, between concepts is called a conceptual relation. In this paper, we will consider only binary relations. A conceptual relation has a direction from one concept, the domain, to another concept, the range. (The domain is sometimes called the sink concept, and the range the source concept.) In a graphical notation, a conceptual relation is represented by an ellipse. Exactly one arrow exits from the ellipse and points to the range. Further, exactly one arrow exiting from the domain points to the ellipse. In fig. 3.1, an example of a conceptual graph is shown. The figure shows that, for example, the domain of AGNT$_{BP}$ is BUYS and its range is PERSON. The parenthesized expressions next to the conceptual relations specify cardinality constraints (as explained below). The generalization constraints TEACHER £ PERSON and STUDENT £ PERSON are represented by arcs labelled ISA.

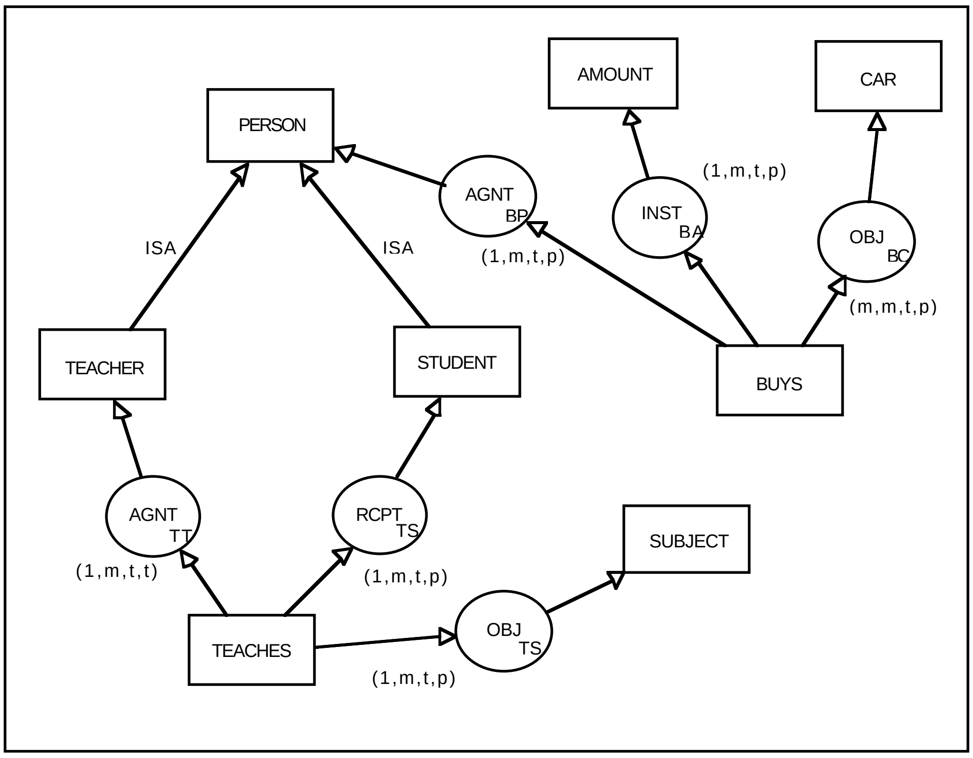

Fig. 3.1 Example of a conceptual graph

One of the theoretical foundations of conceptual graph theory is case grammar. The word “case” has been used to refer to several related concepts. By “surface case” is meant the classification of nouns according to their syntactic roles in a sentence, signalled by various inflected forms. For example, the first person singular pronoun is “I” in nominative case, “me” in accusative case, and “my” in possessive case. Another sense of case, usually called “deep case”, is a categorisation of noun phrases according to the conceptual role they play in the action described by a sentence. An example of a deep case is the agent case: one who performs an action. Approximately, the nominative and the agent case correspond to each other, but it should be noted that there is no simple one-to-one correspondence between surface cases and deep cases, due to the idiosyncrasies of natural language. Most of the work on deep cases has focussed on identifying a small number of conceptual roles that can be used to describe the meaning of any sentence. However, so far no consensus has emerged on such a set of basic cases; instead many different case systems have been proposed, see [Bruce75] for a survey. Examples of deep cases included in most case systems are the following: agent: one who performs an action; object: the one who suffers an action; recipient: the one for whom an action is performed; instrument: an object used to perform an action; location: the place where an action is performed. In conceptual graphs, the concept types correspond to either nouns or verbs, while the conceptual relations correspond to cases. As a consequence, the domain of any conceptual relation is a verb, and the range is a noun.

Thanks to the use of case grammar, conceptual graph theory provides a set of labels for conceptual relations. These labels help designers to choose meaningful and standardized names when modelling object structures. Another advantage of conceptual graphs is that they are associated with a formal mapping to first order logic. However, conceptual graph theory does not provide means for specifying the typical constraints of traditional conceptual modelling approaches. To overcome this limitation, Creasy and Moulin have suggested an extended version of conceptual graph theory in [Creasy92] called ECG (Extended Conceptual Graphs). Basically, the proposal consists of refining some conceptual relation properties within the framework of conceptual graph theory in order to be able to describe semantic structures which are equivalent to those expressible in conceptual modelling approaches. Creasy and Moulin suggest an approach in which conceptual relations are marked with cardinalities. This approach is directly compatible with conceptual modelling approaches and makes it possible to manipulate conceptual structures using the operations applicable to conceptual graphs.

For each conceptual relation, we specify its cardinality constraints, i.e. if it is single-valued, injective, total, or surjective. Informally, a conceptual relation is single-valued if, for each instance of its domain, it has at most one value. A conceptual relation is total if it must have a value for each instance of its domain. A conceptual relation is injective if two different instances of the domain cannot have the same value. A conceptual relation is surjective if each instance of the range is the value of the conceptual relation for some instance of the domain. In the graphical notation, cardinality constraints are represented by a quadruple next to a conceptual relation. The first component of the quadruple indicates whether the conceptual relation is single-valued or not. When single-valued, this component is set to *1*, otherwise *m*. The second component of the structure indicates whether the conceptual relation is injective, in which case the component is set to *1*, otherwise *m*. The third component indicates the totality of the conceptual relation. A *t* indicates totality, a *p* partiality. The fourth component indicates the surjectivity of the conceptual relation. When surjective, this component is set to *t*, otherwise *p*.

We now proceed with a formalization of the ECG-approach outlined above. (Our formalization differs slightly from the one given in [Creasy92].) We first recall some basic definitions regarding first order languages.

Let P and C be two sets of symbols. A *(first order) language based on <P,C>*, written *L(P,C)* is defined on an alphabet consisting of connectives, quantifiers, punctuation symbols, variables, constants C, and predicate symbols P, where each predicate symbol has an arity. A (first order) *formula* in a language L is defined as usual. A *term* is a constant or a variable. An *atom* is a formula of the form $p(t_1,...,t_n)$, where p is a predicate symbol and $t_1,...,t_n$ are terms. A *ground formula* is a formula without variables. For any language L(P,C) we assume that P contains a special symbol, “=”, which is interpreted as the identity relation.

In general, an *integrity constraint* is defined as any closed first order formula. In this paper, however, we only consider certain special cases of constraints, which occur frequently in conceptual modelling. A *typing constraint* is a formula of the form ,x,y(A(x,y) § D(x)) or the form ,x,y(A(x,y) § R(y)). A formula of the first form will be abbreviated “domain(A) = D”, and a formula of the second form as “range(A) = R”. We also say that the domain of A is D and the range of A is R. *Mapping constraints* concern the cardinality of conceptual relations and have one of the following four forms. The expression “the conceptual relation A is single-valued” is an abbreviation of the formula ,x,y,z(A(x,y) ¤ A(x,z) § y = z). The expression “the conceptual relation A is injective” is an abbreviation of the formula ,x,y,z(A(y,x) ¤ A(z,x) § y = z). The expression “the conceptual relation A is total” is an abbreviation of the formula ,x(P(x) § ·yA(x,y)), where P is the domain of A. The expression “the conceptual relation A is surjective” is an abbreviation of the formula ,x(P(x) § ·yA(y,x)), where P is the range of A. A *generalization constraint* is a formula of the form ,x(P(x) § Q(x)) and is abbreviated “P £ Q”. An *exclusion constraint* is a formula of the form ,x(¬(P(x) ¤ Q(x))) and is abbreviated “P ™ Q = Ø”.

A conceptual schema is usually informally defined as an implementation independent description of the contents in an information system. What distinguishes our notion of a conceptual schema from the one found in traditional modelling approaches is mainly the connection to case grammar. For each conceptual relation we specify its case, and for each concept type we specify its category, i.e. noun or verb. In the following, we will assume that there is a set of cases, CAS, from which all schemas draw their conceptual relations. We will not specify the contents of CAS, since our results are valid for any choice of case relations. To specify the cases of conceptual relations and the categories of concept types in a schema, we introduce the notion of a case function; a *case function* for a language L(P, Co) is a function from P to CAS ¡ {Noun, Verb}. To simplify the exposition, we will not give the case functions explicitly in our examples, but instead name the conceptual relations so that the cases become evident.

We now define a *conceptual schema* as a triple <L,IC,C>, where L is a language, IC is a set of typing, mapping, exclusion, and generalization constraints, and C is a case function for L. It is assumed that all predicate symbols in L are unary or binary. In the following, we shall call the unary predicate symbols “concepts” and the binary “conceptual relations”. We assume that for each conceptual relation p, there is a constraint in IC of the form “domain(p) = d”, where C(d) = Verb, and a constraint in IC of the form “range(p) = r”, where C(r) = Noun. Further, C maps the concept types into {Noun, Verb} and the conceptual relations into CAS.

We now turn to the definition of an information base; informally an information base contains information about particular objects and the associations between these. An *information base* for a conceptual schema <L(P,Co), IC, C> is a pair <Cf, F>, where Cf is a finite set of constants, and F is a finite set of ground atoms whose predicate symbols belong to P and whose terms belong to Co ¡ Cf. Note that the information base can be viewed as an interpretation of L over the Herbrand universe Co ¡ Cf.

The role of integrity constraints is to state conditions that must hold for each information base. Let CS = <L,IC,C> be a conceptual schema and IB = <Cf, F> an information base for CS. The information base IB *violates* the schema CS if some constraint in IC is not true in F.

An example of a small information base for the schema in fig. 3.1 is the following. Note that the information base does not violate the schema.

{teacher(t1), student(s1), subject(b1), teaches(c1), agnt_{tt}(c1,t1), rcpt_{ts}(c1,s1), obj_{ts}(c1,b1)}

4 Integration Assertions and Conflictfreeness

In this section, we introduce a number of basic types of integration assertions. Integration assertions are used to describe correspondences between constructs in different conceptual schemas with corresponding information bases. The assertions introduced here describe set relationships between the extensions of different predicate symbols, such as the extension of one predicate symbol being identical to or included in the extension of another. Another important type of integration assertions are object equality assertions (as identified in e.g. [Johannesson91]). These express that different terms in different information bases denote the same object. However, to simplify the exposition we will not consider object equality assertions in this paper, and we make the assumption that equal terms from different information bases denote the same object.

Let CS = <L(P,Co), IC, C> and CS' = <L'(P',Co'), IC', C'> be two conceptual schemas. Let E (E') and F (F') be two object types of CS (CS'), and let a (a') be a conceptual relation of CS (CS') with E (E') as domain and F (F') as range, see fig. 4.1. We introduce two basic types of integration assertions:

(1) Concept type inclusion:

,x(E(x) § E'(x))

(2) Conceptual relation equality:

,x,y(E(x) ¤ E'(x) ¤ F(y) ¤ F'(y) § (a(x,y) ¶ a'(x,y)))

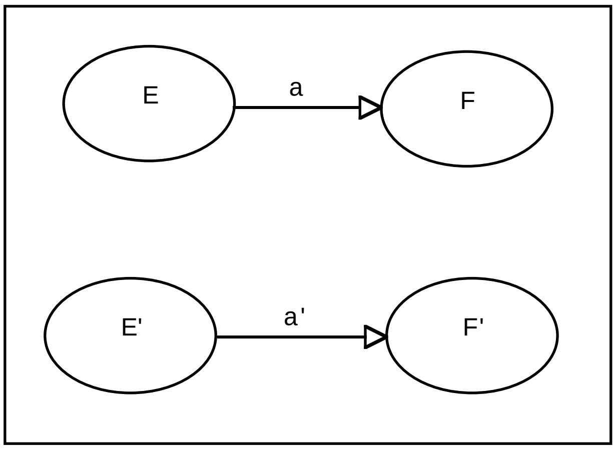

Fig. 4.1 Two simple schemas

Assertions of type 1 express that the extension of one type is included in that of another, while expressions of type 2 state that the extension of one conceptual relation is equal to that of another. We introduce some abbreviations for these integration assertions. Formula (1) is abbreviated E £ E', and formula (2) is abbreviated a = a'. We write E = E' iff E £ E' and E' £ E. An *integration specification* is a set of integration assertions of type (1) above.

When specifying the correspondences between two schemas, it is sufficient to give a set of concept type inclusions, i.e. an integration specification. The conceptual relation equalities can be derived in the following way. If there is a concept type inclusion between two types, then there shall be a conceptual relation equality between those pairs of conceptual relations that have the two types as domains and have the same case (according to the case functions C and C'). Thus, the use of cases in the ECG formalism simplifies the specification of integration assertions. To express formally how conceptual relation equalities can be derived, we define a function *f* that takes a concept type inclusion as argument and returns a set of conceptual relation equalities. Let A = ,x(E(x) § E'(x)) be an integration assertion of type 1. We define ƒ(A) = {,x,y(E(x) ¤ E'(x) ¤ F(y) ¤ F'(y) § (a(x,y) ¶ a'(x,y))) | a ´ P, a' ´ P', domain(a) = E, domain(a') = E', range(a) = F, range(a') = F', C(a) = C'(a')}.

In fig. 4.2, two conceptual schemas are shown. Let IS be the following integration specification:

PERSON £ HUMAN
TEACHER £ PROFESSOR
UNDERGR. STUDENT £ STUDENT
PET £ ANIMAL
TOPIC £ SUBJECT
POSSESSES £ OWNS
LECTURES £ TEACHES

The following conceptual relation equalities can be derived from IS, that is the set ¡A´IS ƒ(A):

$AGNT_{LP} = AGNT_{TT}$
$RCPT_{LU} = RCPT_{TS}$
$OBJ_{LT} = OBJ_{TS}$
$AGNT_{PP} = AGNT_{OH}$
$OBJ_{PP} = OBJ_{OA}$

Given two conceptual schemas and a set of integration assertions, it is possible that these are, in some sense, conflicting. Informally, we say they are conflicting if there exists an information base for one of the schemas, and there does not exist a corresponding information base for the other schema such that the integration assertions hold. This idea is formalized in the following definition:

Let CS1 and CS2 be two conceptual schemas and IA a set of integration assertions for CS1 and CS2. *CS2 is conflictfree w.r.t. CS1 and IA* if for every non-violating information base IB1 = <C1,F1> for CS1, there exists a non-violating information base IB2 = <C2,F2> for CS2, such that IA is true in F1 ¡ F2.

Based on the concept of conflictfreeness, we introduce the notion of two schemas being in conflict. Let CS1, CS2, and IA be as above. *CS2 conflicts with CS1 given IA* if CS2 is not conflictfree w.r.t. CS1 and IA.

Checking for conflicts is an important means for validating that a set of integration assertions is reasonable. However, it can be shown that in the presence of general integrity constraints (i.e. not only those introduced above), the problem of conflictfreeness is undecidable; see [Convent86], where a similar definition of conflictfreeness is given in the context of the relational model. This means that the best one can hope to achieve is to identify necessary and/or sufficient conditions for conflictfreeness in special cases.

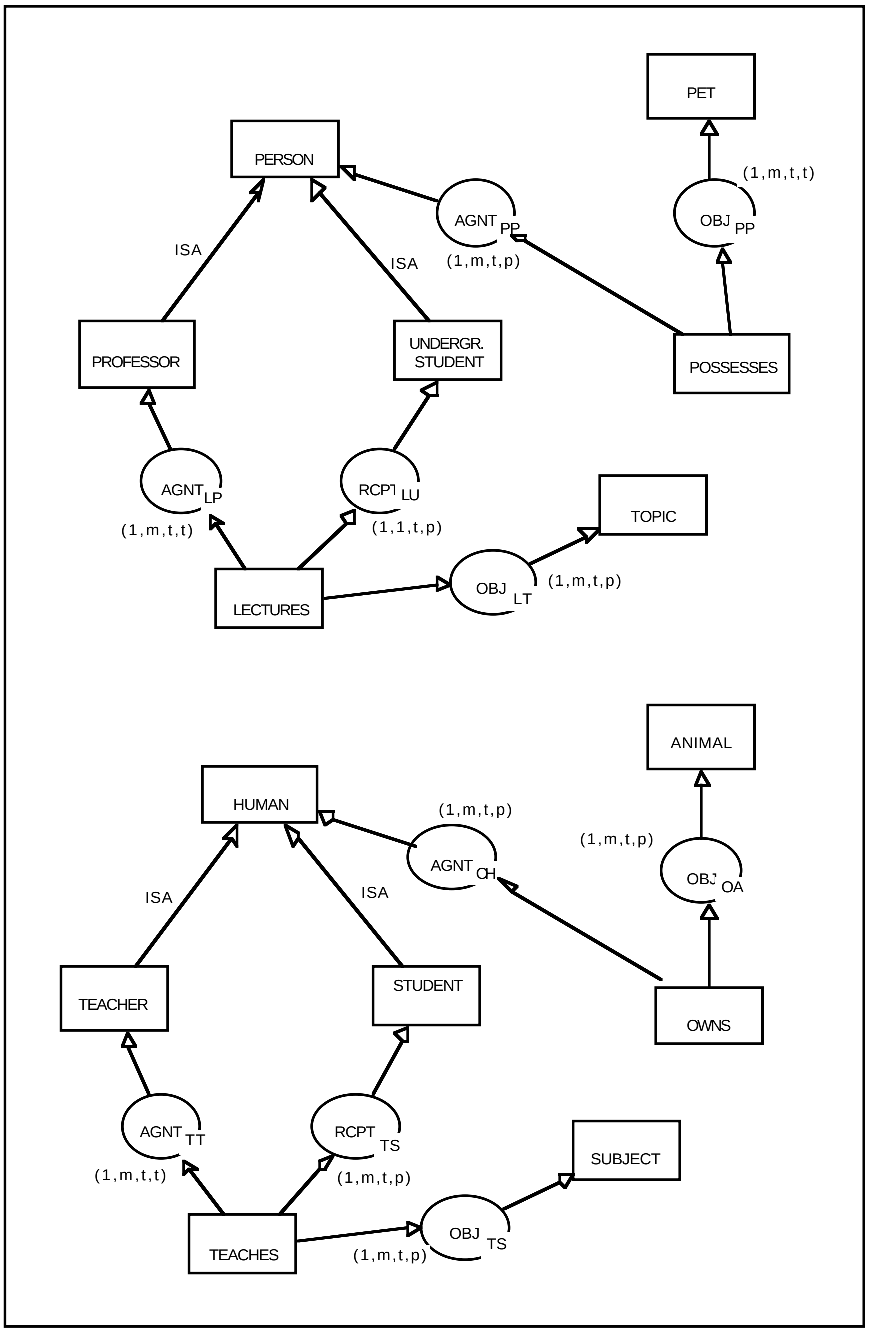

Fig. 4.2 Example of two conceptual schemas

In the remainder of this section, we will discuss necessary conditions for two schemas to be conflictfree, given a set of integration assertions. At first inspection, one may be tempted to believe that a necessary condition for conflictfreeness is that if two conceptual relations of two different schemas are asserted to be equal, then the same constraints must apply to them. However, there are cases where two schemas are conflictfree although different constraints apply to equal conceptual relations. This is illustrated in the following examples.

Consider the schemas in fig. 4.2 and the integration assertions above. One may be lead to believe that the schemas are conflicting, since the conceptual relation $RCPT_{LU}$ is injective, whereas $RCPT_{TS}$ is not injective. This would indeed have been the case if the range of $RCPT_{LU}$ had been equal to the range of $RCPT_{TS}$. In the example above, however, the range of $RCPT_{LU}$ is only included in the range of $RCPT_{TS}$, which entails that the different constraints on $RCPT_{LU}$ and $RCPT_{TS}$ do not make the schemas conflicting. This can easily be seen if we suppose that UNDERGR. STUDENT contains only students studying one subject at a time, whereas STUDENT also contains students taking several courses simultaneously.

As another example, consider the conceptual relations OBJ_{PP} and OBJ_{OA} in fig. 4.2, which are asserted to be equal. Although only the conceptual relation OBJ_{PP} is surjective, the schemas are not necessarily conflicting. Since the range of OBJ_{PP} is only included in the range of OBJ_{OA}, it may be the case that those animals which are not owned by anyone are not pets, and thus the schemas do not conflict. We now give a number of sufficient conditions for two schemas to be conflicting given a set of integration assertions.

Let CS1 = <L1, IC1, C1> and CS2 = <L2, IC2, C2> be two conceptual schemas, and let IA be a set of integration assertions for CS1 and CS2. An exclusion constraint A ™ B = Ø in IC1 is *conflict generating* w.r.t. CS2 and IA if there are concept types D1 and D2 in L1 and L2, respectively, such that D1 £ A and D2 £ B follow from IC1 ¡ IC2, and D1 £ D2 or D2 £ D1 follows from IA.

In the following, let a and a' be conceptual relations in L1 and L2, respectively. Let E and E' be the domains, and F and F' the ranges of a and a', respectively (see fig. 4.1).

The constraint “a is single-valued” in IC1 is *conflict generating* w.r.t. CS2 and IA if “a' is single-valued” does not belong to IC2, and a = a', E' £ E, and F' £ F follow from IA.

The constraint “a is injective” in IC1 is *conflict generating* w.r.t. CS2 and IA if “a' is injective” does not belong to IC2, and a = a', E' £ E, and F' £ F follow from IA.

The constraint “a is total” in IC1 is *conflict generating* w.r.t. CS2 and IA if “a' is total” does not belong to IC2, and a = a', E' £ E, and F £ F' follow from IA.

The constraint “a is surjective” in IC1 is *conflict generating* w.r.t. CS2 and IA if “a' is surjective” does not belong to IC2, and a = a', E £ E', and F' £ F follow from IA.

We now summarize a number of sufficient conditions for conflict in the following proposition, which is straight-forward to prove.

Proposition: Let CS1 = <L1, IC1, C1> and CS2 = <L2, IC2, C2> be two conceptual schemas. Let IA be a set of integration assertions for CS1 and CS2. CS1 conflicts with CS2 given IA if IC1 contains a constraint that is conflict generating w.r.t. CS2 and IA.

5 Schema Compatibility

In this section, we will discuss how the distinguishing features of the ECG formalism can be utilized to facilitate the schema comparison step in schema integration. The basic idea is to increase the similarity between the schemas to be integrated by expanding them in two different ways. First, the schemas shall be extended by connecting them to a common terminology, which is defined in a concept taxonomy. Secondly, the schemas shall be expanded by systematically adding conceptual relations in such a way that all possible case relations are included.

We first introduce the notion of a taxonomy structure. A *taxonomy structure* is a triple <L(P,Co), IC, C>, where P is a set of concepts, IC is a set of exclusion and generalization constraints, and C is a case function as defined above. Thus, a taxonomy structure is a conceptual schema with a simple form. We envisage a taxonomy structure to contain a number of generic concepts for a UoD. The purpose of the taxonomy structure is to provide a standardized terminology, which can be used as a common framework for the schemas to be integrated.

Two schemas describing the same UoD can look very different from each other if they focus on different aspects of the phenomena in the UoD. One way to increase the similarity of such schemas is to systematically add conceptual relations to them. Intuitively, the idea is that for each concept type that is classified as a verb (by the case function of the schema), conceptual relations corresponding to all cases that are applicable to that verb are added. We formalize this idea through the notion of a complementary schema.

Let S = <L(P,Co), IC, C> be a conceptual schema. Let T = <Lt(Pt,Cot), ICt, Ct> be a taxonomy structure. Let V = {c ´ P | C(c) = Verb}. Let rel: V § Con be a function, where Con is a set of conceptual relations and Con ™ P = Ø. Let const: V § 2^{Constr} be a function, where Constr is the set of all integrity constraints expressible in L(P ¡ Con, Co). Let CR = ¡$_{\text{c´V}}${rel(c)}. Let Ce: P ¡ CR § CAS ¡ {Verb, Noun} be an extension of C. Let GC be a set of generalization constraints of the form c £ t, where c is a concept type in P, and t is a concept type in Pt. A *complementary schema* of S given T is a schema <L', IC', C'>, where L' = L ¡ Lt ¡ CR, IC' = IC ¡ GC ¡ ICt ¡$_{\text{c´V}}$ const(c), and C' = Ce ¡ Ct. (It is assumed that rel is chosen so that it returns as many conceptual relations as possible for each verb in V, and analogously for Constr and GC.)

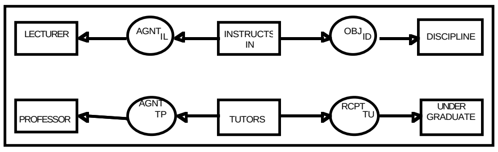

Fig. 5.1 Two schemas focusing on different aspects of the same UoD

The purpose of constructing a complementary schema is to be able to extend the original schema in a systematic way and connect it to the concepts in the taxonomy structure. Complementary schemas are useful during schema comparison in that they reduce schema dissimilarities caused by differences in terminology and focus. Even if the original schemas use different terminologies to describe the same UoD, the complementary schemas will use the common terminology specified in the taxonomy structure. In cases where two schemas model the same phenomenon but differ in focus, their complementary schemas will be more similar as a result of the systematic extension with additional conceptual relations. As an example, consider the two schemas in fig. 5.1. Both schemas model persons who teach but look different since they use different terminology and focus on different aspects. The first schema models which subjects a person teaches, while the second describes which students a teacher has. In the complementary schemas of fig. 5.2, these differences in focus have disappeared, and a common terminology has been introduced.

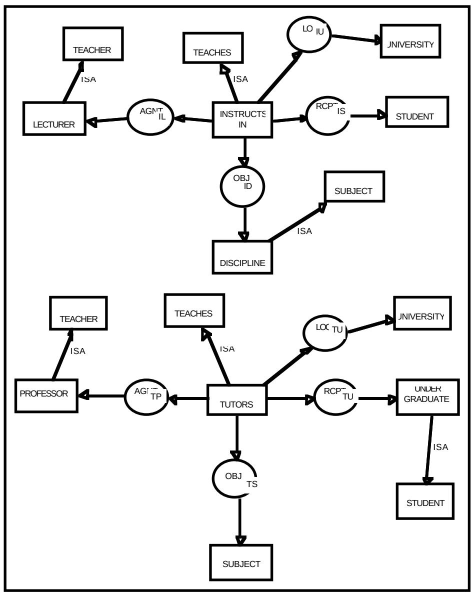

Fig. 5.2. The complementary schemas of the schemas in fig. 5.1

We conclude this section with a condition specifying when two schemas can be meaningfully integrated. The condition takes into account not only the original schemas to be integrated but also their complementary schemas. The condition is therefore stronger than the requirement that the originals schemas be conflictfree.

Let S1 and S2 be two schemas and T a taxonomy structure. Let CS1 = <L1, IC1, C1> and CS2 = <L2, IC2, C2> be the complementary schemas of S1 and S2 given T, respectively. Let IS be an integration specification for CS1 and CS2, and let IA = IS ¡A´IS ƒ(A). S1 and S2 are *compatible w.r.t. T and IA* if the following three conditions are satisfied:

- CS1 is conflictfree w.r.t. CS2 and IA
- CS2 is conflictfree w.r.t. CS1 and IA

- For each A = ,x(E(x) § E'(x)) in IS, ©CSi(E) £ ©CS2j(E'), where ©CSk(c) = {Ck(a) | domain(a) = c}, where E belongs to Li and E' to Lj, i ≠ j.

6 Summary and Conclusions

In this paper, we have attempted to show how the expressiveness of a semantically rich modelling formalism can be used to support the schema integration process. The distinguishing feature of the modelling formalism chosen, ECG, is its use of case grammar. Describing all associations between concepts by a small number of case relations standardizes the terminology of the schemas, and thus facilitates schema comparison. Another advantage of case grammar is that it provides a framework for systematically extending the schemas to be integrated in order to increase their similarity to each other. We have also seen how the ECG formalism facilitates the specification of schema correspondences by so called integration specifications. Finally, we have given two conditions expressing when schemas can be meaningfully integrated: conflictfreeness and compatibility.

The results outlined in this paper can be utilized in the schema integration process in two different ways. One alternative is that a user first proposes a set of integration assertions, and after that one verifies that the conditions of conflictfreeness and compatibility are satisfied. A more ambitious alternative would be to let a computer based tool suggest an integration specification satisfying the conditions stated above. A human user would then be required to approve of the suggested specification before it could be used in the subsequent steps of the schema integration process.

It is important, when constructing a schema in the ECG formalism, to decide upon an appropriate set of cases. In the literature, there are many proposals for different case grammar systems, [Bruce75], and which one to choose in a specific situation depends on the characteristics of the UoD being modelled. A similar problem is knowing how to construct an appropriate taxonomy structure. Obviously, the larger the taxonomy structure, the more useful it will be for schema comparison. On the other hand, a large taxonomy structure becomes difficult to construct and maintain.

The question remains how practical it is to follow the approach outlined in this paper. It might be argued that using the ECG formalism and constructing taxonomy structures and complementary schemas is too complicated a process if the goal is just to compare and integrate a small number of conceptual schemas. One answer to this is that expressing schemas in the ECG formalism also provides many other advantages beyond facilitating schema integration. For example, ECG schemas are able to express more of the semantics of an application than schemas in traditional modelling languages, and the use of case relations makes them adequate as the basis of natural language interfaces to databases. Constructing taxonomy structures and complementary schemas may not be worthwhile if only a few schemas are to be integrated at a single occasion. In many contexts, however, it is required that a large number of schemas be compared over an extended period of time, e.g. in federated information systems [Sheth90].

References

[Batini86] C. Batini, M. Lenzerini and S. B. Navathe, “A Comparative Analysis of Methodologies for Database Schema Integration”, *ACM Computing Surveys*, vol. 18, no. 4, pp. 323-364, 1986.

[Beeri89] C. Beeri, “Formal Models for Object Oriented Databases”, in *First International Conference on Deductive and Object Oriented Databases*, Ed. W. Kim, pp. 405-430, Kyoto, North-Holland, 1989.

[Biskup86] J. Biskup and B. Convent, “A Formal View Integration Method”, in *International Conference on the Management of Data,* Ed. pp. Washington, ACM, 1986.

[Bouzeghoub90] M. Bouzeghoub and I. Comyn-Wattiau, “View Integration by Semantic Unification and Transformation of Data Structures”, in *Ninth International Conference on Entity-Relationship Approach,* Ed. H. Kangassalo, pp. 413-430, Lausanne, North-Holland, 1990.

[Bright90] W. Bright and A. Hurson, “A Taxonomy and Current Issues in Multidatabase Systems”, *IEEE Computer*, vol. 24, no. 10, 1990.

[Bruce75] B. Bruce, “Case Systems for Natural Language”, *Artificial Intelligence*, vol. 6, pp. 327-360, 1975.

[Chen76] P. P. Chen, “The Entity-Relationship Model - Toward a Unified View of Data”, *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9-36, 1976.

[Collet91] C. Collet, M. Huhns and W. Shen, “Resource Integration Using a Large Knowledge Base in Carnot”, *IEEE Computer*, pp. 55-62, December 1991.

[Convent86] B. Convent, “Unsolvable Problems Related to the View Integration Approach”, in *International Conference on Database Theory*, Ed. pp. 141-156, Rome, 1986.

[Creasy92] P. Creasy and B. Moulin, “Extending the Conceptual Graph Approach for Data Conceptual Modelling”, *Data and Knowledge Engineering*, vol. 8, no. 3, pp. 223-248, 1992.

[ElMasri85] R. ElMasri, J. Weeldryer and A. Hevner, “The Category Concept: An Extension to the Entity-Relationship Model”, *Data and Knowledge Engineering*, vol.1, no.1, 1985

[Gallaire84] H. Gallaire, J. Minker and J. M. Nicholas, “Logic and Databases: A Deductive Approach”, *ACM Computing Surveys*, vol. 16, no. 2, pp. 1984.

[Guha90] R. Guha and D. Lenat, “CYC: A Midterm Report”, *AI Magazine*, Fall 1990

[Hammer81] M. Hammer and D. McLeod, “Database Description with SDM: A Semantic Database Model”, *ACM Transactions on Database Systems*, vol.6, no.3, pp. 351-386, 1981.

[Hull87] R. Hull and R. King, “Semantic Database Modeling: Survey, Applications and Research Issues”, *ACM Computing Surveys*, vol.19, no.3, pp. 201-260, 1987.

[ISO82] E. J. J. v. Griethuysen, “ISO - Concepts and Terminology for the Conceptual Schema and the Information Base”, N695, ISO/TC9/SC5/WG3, 1982.

[Johannesson91] P. Johannesson, “A Logic Based Approach to Schema Integration”, in *10th International Conference on Entity-Relationship Approach,* Ed. T. Teorey, San Fransisco, North-Holland, 1991.

[Johannesson93a] P. Johannesson, “A Logical Basis for Schema Integration”, in *Third International Workshop on Research Issues in Data Engineering - Interoperability in Multidatabase Systems,* Ed. H. Schek, Vienna, IEEE Press, 1993.

[Johannesson93b] P. Johannesson, “Schema Transformations as an Aid in View Integration”, in *5th International Conference on Computer Aided Information Systems Engineering,* Ed. C. Rolland, Paris, Springer, 1993.

[Johannesson93c] P. Johannesson, *Schema Integration, Schema Translation, and Interoperability in Federated Information Systems*, PhD thesis, Department of Computer and Systems Sciences, Stockholm University, 1993.

[Larson89] J. A. Larson, S. Navathe and R. ElMasri, “A Theory of Attribute Equivalence in Databases with Apllications to Schema Integration”, *IEEE Transactions on Software Engineering*, vol. 15, no. 4, pp. 449-463, 1989.

[Lloyd87] J. Lloyd, *Foundations of Logic Programming*, Springer Verlag, 1987.

[Motro87] A. Motro, “Superviews: Virtual Integration of Multiple Databases”, *IEEE Transactions on Software Engineering*, vol. 13, no. 7, pp. 785-798, 1987.

[Nijssen89] G. Nijssen and T. Halpin, *Conceptual Schema and Relational Database Design*, Prentice-Hall 1989.

[Peckham88] J. Peckham and F. Maryanski, “Semantic Data Models”, *ACM Computing Surveys*, vol.20, no.3, pp. 153-190, 1988.

[Sheth90] A. P. Sheth and J. A. Larson, “Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases”, *ACM Computing Surveys*, vol. 22, no. 3, pp. 183-236 1990.

[Song92] W. W. Song, P. Johannesson and J. A. Bubenko Jr, “Semantic Similarity Relations in Schema Integration”, in *11th International Conference on the Entity-Relationship Approach,* Ed. A. M. Tjoa, Karlsruhe, Germany, 1992.

[Sowa84] J. F. Sowa, Conceptual Structures - Information Processing in Mind and Machine, Addison-Wesley, 1984.

[Spaccapietra92] S. Spaccapietra, C. Parent and Y. Dupont, “Model Independent Assertions for Integration of Heterogeneous Schemas”, *The VLDB Journal*, vol. 1, no. 2, pp. 81-126, 1992.